## EAST SEARCH TAGGED

| patent # | type | pub date | title | class1 | class2 | inventor | | | |
|---|---|---|---|---|---|---|---|---|---|
| US 5049985 A | USPAT | 19910917 | Color images reading apparatus having transformation table formed base on average values of plural color component signals | 358/530 | 358/515 | Outa; Ken-Ichi | | | |
| US 5296945 A | USPAT | 19940322 | Video ID photo printing apparatus and complexion converting apparatus | 358/518 | 358/501; 358/523; 358/538 | Nishikawa; Masaji et al. | | | |
| US 5539523 A | USPAT | 19960723 | Image forming device making color corrections on stored image data | 358/296 | 358/504; 358/518; 358/527; 382/118; 382/165; 382/190 | Nakai; Yoshiyuki et al. | | | |
| US 6396599 B1 | USPAT | 20020528 | Method and apparatus for modifying a portion of an image in accordance with colorimetric parameters | 358/1.9 | 358/518; 382/164 | Patton; David L. et al. | | | |
| US 6535301 B1 | USPAT | 20030318 | Image processing apparatus, image processing method, image processing program recording medium, color adjustment method, color adjustment device, and color adjustment control program recording medium | 358/1.9 | 358/520 | Kuwata; Naoki et al. | | | |
| US 6643398 B2 | USPAT | 20031104 | Image correction device, image correction method and computer program product in memory for image correction | 382/167 | 358/1.9; 358/3.1; 382/168 | Moriwaki; Kagumi | | | |
| US 6650771 B1 | USPAT | 20031118 | Color management system incorporating parameter control channels | 382/162 | 382/302 | Walker; Douglas G. | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| US 6717698 B1 | USPAT | 20040406 | Tone scale processing based on image modulation activity | 358/1.9 | 358/3.23 | Lee; Hsien-Che | | | |
| US 6791716 B1 | USPAT | 20040914 | Color image reproduction of scenes with preferential color mapping | 358/1.9 | 358/518; 358/520; 382/162; 382/167 | Buhr; John D. et al. | | | |
| US 20010035988 A1 | US-PGPUB | 20011101 | Color image processing apparatus and method, and computer- readable recording medium in which color image processing program is recorded | 358/518 | | Semba, Satoshi et al. | | | |

| | | |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## EAST SEARCH HISTORY

| HITS | QUERY | DATABASE | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 23 | (US-20010005222-$ or US-20020172419-$ or US-20030152283-$).did. or (US-5130789-$ or US-5130935-$ or US-5384601-$ or US-5528339-$ or US-5539523-$ or US-5583666-$ or US-5828779-$ or US-5828780-$ or US-6141431-$ or US-6148092-$ or US-6269184-$ or US-6272239-$ or US-6377702-$ or US-6587225-$ or US-6594388-$ or US-6650771-$ or US-6670963-$ or US-6678407-$ or US-6690822-$ or US-6748097-$).did. | US-PGPUB; USPAT | | | | | | |
| 2 | 20030012414 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 2 | 5486853.pn. | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 2 | 5301344.pn. | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 125 | (((flesh or skin) near (tone or color or colour or pigment or chroma$6 or tint or hue or complexion) and ("382".clas. or "358".clas.)) and (map or mapping or mapped or (transfer adj1 function) or mtf or gamut)) and @ad<"20010702" and ((detect$6 or segment$5 or recogn$6 or identif$6) same (face or head or facial)) | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 29 | S137 and ((correct$5 or enhanc$5 or improv$5) near3 (tone or color or colour or pigment or chroma$6 or tint or hue or complexion)) | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 180 | ((correct$5 or enhanc$5 or improv$5) near3 (tone or color or colour or pigment or chroma$6 or tint or hue or complexion) near3 (flesh or skin)) and ("382".clas. or "348".clas. or "358".clas.) and @ad<"20010629" | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 234 | ((correct$5 or enhanc$5 or improv$5) near5 (tone or color or colour or pigment or chroma$6 or tint or hue or complexion) near5 (flesh or skin)) and ("382".clas. or "348".clas. or "358".clas.) and @ad<"20010629" | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 41558 | (map or mapping or mapped or (transfer adj1 function) or mtf or gamut or curve or function) near3 (tone or color or colour or pigment or chroma$6 or tint or hue or complexion) | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 97 | S140 and S141 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 21 | S140 and S141 and (histogram) | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 21 | S140 and S141 and (histogram) and (flesh or skin) | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 14 | ("5162899" \| "5418895" \| "5424945" \| "5694484" \| "5751854" \| "5796428" \| "5835627" \| "5875265" \| "6282317" \| "6346998" \| "6389169" \| "6522418" \| "6529630" \| "6563596").PN. | US-PGPUB; USPAT; USOCR | | | | | | |
| 12 | ("5539523").URPN. | USPAT | | | | | | |
| 22 | ("4007362" \| "4929978" \| "4937662" \| "5018008" \| "5049985" \| "5132788" \| "5136401" \| "5142355" \| "5440409" \| "5442716").PN. OR ("5539523").URPN. | US-PGPUB; USPAT; USOCR | | | | | | |
| 2 | US-6130741-$.DID. | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 2 | US-6792130-$.DID. | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 30 | (flesh or skin) with (tone near (correct$5 or reproduc$5 or enhanc$5)) and ("382" or "358" or "345").clas. | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 14 | (flesh or skin) with (tone near (correct$5 or reproduc$5 or enhanc$5)) and ("382" or "358" or "345").clas. and @ad<"20010629" | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 93 | (flesh or skin) with ((tone or color or colour) near (correct$5 or reproduc$5 or enhanc$5)) and ("382" or "358" or "345").clas. and @ad<"20010629" | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 89 | (flesh or skin) with ((tone or color or colour) near (correct$5 or reproduc$5 or enhanc$5)) and ("382" or "358" or "345").clas. and @ad<"20010629" and (not S155) | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 79 | (flesh or skin) with ((tone or color or colour) near (correct$5 or reproduc$5 or enhanc$5)) and ("382" or "358" or "345").clas. and @ad<"20010629" and (not S154) | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 79 | (US-20030053085-$ or US-20030016376-$ or US-20020163657-$ or US-20020030831-$ or US-20020005965-$ or US-20010053247-$ or US-20010043368-$ or US-20010035989-$ or US-20010035988-$ or US-20010016064-$ or US-20010005222-$).did. or (US-6868303-$ or US-6853747-$ or US-6750890-$ or US-6738510-$ or US-6710892-$ or US-6694051-$ or US-6678407-$ or US-6668079-$ or US-6633410-$ or US-6603878-$ or US-6553140-$ or US-6549653-$ or US-6535301-$ or US-6480300-$ or US-6459500-$ or US-6384837-$ or US-6377702-$ or US-6330076-$ or US-6292574-$ or US-6278533-$ or US-6272239-$ or US-6262778-$ or US-6256414-$ or US-6252976-$ or US-6229580-$ or US-6169536-$).did. or (US-6160912-$ or US-6101272-$ or US-6023351-$ or US-6008812-$ or US-5986718-$ or US-5960162-$ or US-5949427-$ or US-5946412-$ or US-5930388-$ or US-5875036-$ or US-5874988-$ or US-5870491-$ or US-5867286-$ or US-5852675-$ or US-5675717-$ or US-5631749-$ or US- | US-PGPUB; USPAT | | | | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 79 | (flesh or skin) with ((tone or color or colour) near (correct$5 or reproduc$5 or enhanc$5)) and ("382" or "358" or "345").clas. and @ad<"20010629" and S159 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 30 | ("3739078" \| "4805223" \| "4847677" \| "5027420" \| "5029312" \| "5212518" \| "5225900" \| "5296884" \| "5296945" \| "5300974" \| "5390381" \| "5444487" \| "5447811" \| "5478238" \| "5488429" \| "5528339" \| "5638136" \| "5710654" \| "5715377" \| "5726737" \| "5797750" \| "5815244" \| "6207874" \| "6208749" \| "6215893" \| "6272239" \| "6278533" \| "6293284").PN. OR ("6396599").URPN. | US-PGPUB; USPAT; USOCR | | | | | | |
| 1 | (US-5539523-$).did. | USPAT | | | | | | |
| 1 | S161 and fac$4 | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 1060520 | 1 and digital | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |
| 1 | (US-5539523-$).did. | USPAT | | | | | | |
| 1 | S164 and digital | US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB | | | | | | |

## EAST SEARCH TAGGED

| patent # | type | pub date | title | class1 | class2 | inventor | | | |
|---|---|---|---|---|---|---|---|---|---|
| US 5049985 A | USPAT | 19910917 | Color images reading apparatus having transformation table formed base on average values of plural color component signals | 358/530 | 358/515 | Outa; Ken-Ichi | | | |
| US 5296945 A | USPAT | 19940322 | Video ID photo printing apparatus and complexion converting apparatus | 358/518 | 358/501; 358/523; 358/538 | Nishikawa; Masaji et al. | | | |
| US 5539523 A | USPAT | 19960723 | Image forming device making color corrections on stored image data | 358/296 | 358/504; 358/518; 358/527; 382/118; 382/165; 382/190 | Nakai; Yoshiyuki et al. | | | |
| US 6396599 B1 | USPAT | 20020528 | Method and apparatus for modifying a portion of an image in accordance with colorimetric parameters | 358/1.9 | 358/518; 382/164 | Patton; David L. et al. | | | |
| US 6535301 B1 | USPAT | 20030318 | Image processing apparatus, image processing method, image processing program recording medium, color adjustment method, color adjustment device, and color adjustment control program recording medium | 358/1.9 | 358/520 | Kuwata; Naoki et al. | | | |
| US 6643398 B2 | USPAT | 20031104 | Image correction device, image correction method and computer program product in memory for image correction | 382/167 | 358/1.9; 358/3.1; 382/168 | Moriwaki; Kagumi | | | |
| US 6650771 B1 | USPAT | 20031118 | Color management system incorporating parameter control channels | 382/162 | 382/302 | Walker; Douglas G. | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| US 6717698 B1 | USPAT | 20040406 | Tone scale processing based on image modulation activity | 358/1.9 | 358/3.23 | Lee; Hsien-Che | | | |
| US 6791716 B1 | USPAT | 20040914 | Color image reproduction of scenes with preferential color mapping | 358/1.9 | 358/518; 358/520; 382/162; 382/167 | Buhr; John D. et al. | | | |
| US 20010035988 A1 | US-PGPUB | 20011101 | Color image processing apparatus and method, and computer- readable recording medium in which color image processing program is recorded | 358/518 | | Semba, Satoshi et al. | | | |

# *Miscellaneous Search and Discovery*

US006535301B1

(12) **United States Patent**
**Kuwata et al.**

(10) **Patent No.: US 6,535,301 B1**
(45) **Date of Patent: Mar. 18, 2003**

(54) **IMAGE PROCESSING APPARATUS, IMAGE PROCESSING METHOD, IMAGE PROCESSING PROGRAM RECORDING MEDIUM, COLOR ADJUSTMENT METHOD, COLOR ADJUSTMENT DEVICE, AND COLOR ADJUSTMENT CONTROL PROGRAM RECORDING MEDIUM**

(75) Inventors: **Naoki Kuwata**, Nagano-ken (JP); **Yoshihiro Nakami**, Nagano-ken (JP)

(73) Assignee: **Seiko Epson Corporation**, Tokyo (JP)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/097,828**

(22) Filed: **Jun. 16, 1998**

(30) **Foreign Application Priority Data**

Jun. 17, 1997 (JP) ..... 9-160158
Nov. 13, 1997 (JP) ..... 9-312126

(51) **Int. Cl.**[7] ..... **G06F 15/00**; G03F 3/08
(52) **U.S. Cl.** ..... **358/1.9**; 358/520
(58) **Field of Search** ..... 358/1.9, 1.1, 504, 358/518, 515, 520, 523, 527, 531, 537; 382/167, 162

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | | |
|---|---|---|---|---|
| 4,975,861 | A | 12/1990 | Fujimoto | 364/521 |
| 5,012,333 | A * | 4/1991 | Lee et al. | 358/80 |
| 5,105,267 | A | 4/1992 | Hayashi | 358/80 |
| 5,130,935 | A * | 7/1992 | Takiguchi | 364/526 |
| 5,187,570 | A * | 2/1993 | Hibi et al. | 358/80 |
| 5,289,295 | A * | 2/1994 | Yumiba et al. | 358/518 |
| 5,497,431 | A | 3/1996 | Nakamura | 382/162 |
| 5,717,839 | A * | 2/1998 | Ichikawa | 395/109 |
| 5,754,222 | A * | 5/1998 | Daly et al. | 348/184 |

FOREIGN PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| EP | 0 354 490 | 2/1990 | H04N/1/46 |
| EP | 0 377 386 | 7/1990 | H04N/1/40 |
| EP | 0 441 558 | 8/1991 | G06F/15/72 |
| EP | 0 481 525 | 4/1992 | H04N/1/46 |
| EP | 0 723 364 | 7/1996 | H04N/1/60 |

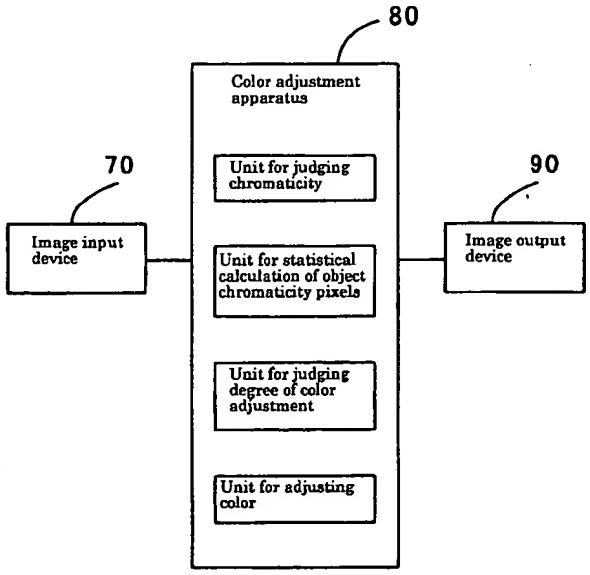
* cited by examiner

*Primary Examiner*—Madeleine Nguyen
(74) *Attorney, Agent, or Firm*—Sughrue Mion, PLLC

(57) **ABSTRACT**

A color adjustment device performs color separation of color image data for each predetermined element color and enhances the color image data to provide each desired color in the output. A chromaticity judging unit determines a chromaticity value of each pixel according to the color image data. An object chromaticity pixel statistical calculation unit performs statistical calculations on pixels having chromaticity values determined by the chromaticity judging unit to meet a predetermined chromaticity range. A color adjustment degree judging unit determines a degree of color adjustment to eliminate a difference between a predetermined optimum value for pixels that meet the predetermined chromaticity range, and a result value attained in the statistical calculation. The color adjustment degree judging unit also regulates the degree of color adjustment according to an occupancy ratio, of pixels subjected to the statistical calculation to the total number of pixels. A color adjusting unit carries out the color adjustment of the image data according to the regulated degree of color adjustment.

**6 Claims, 30 Drawing Sheets**



# Fig. 11

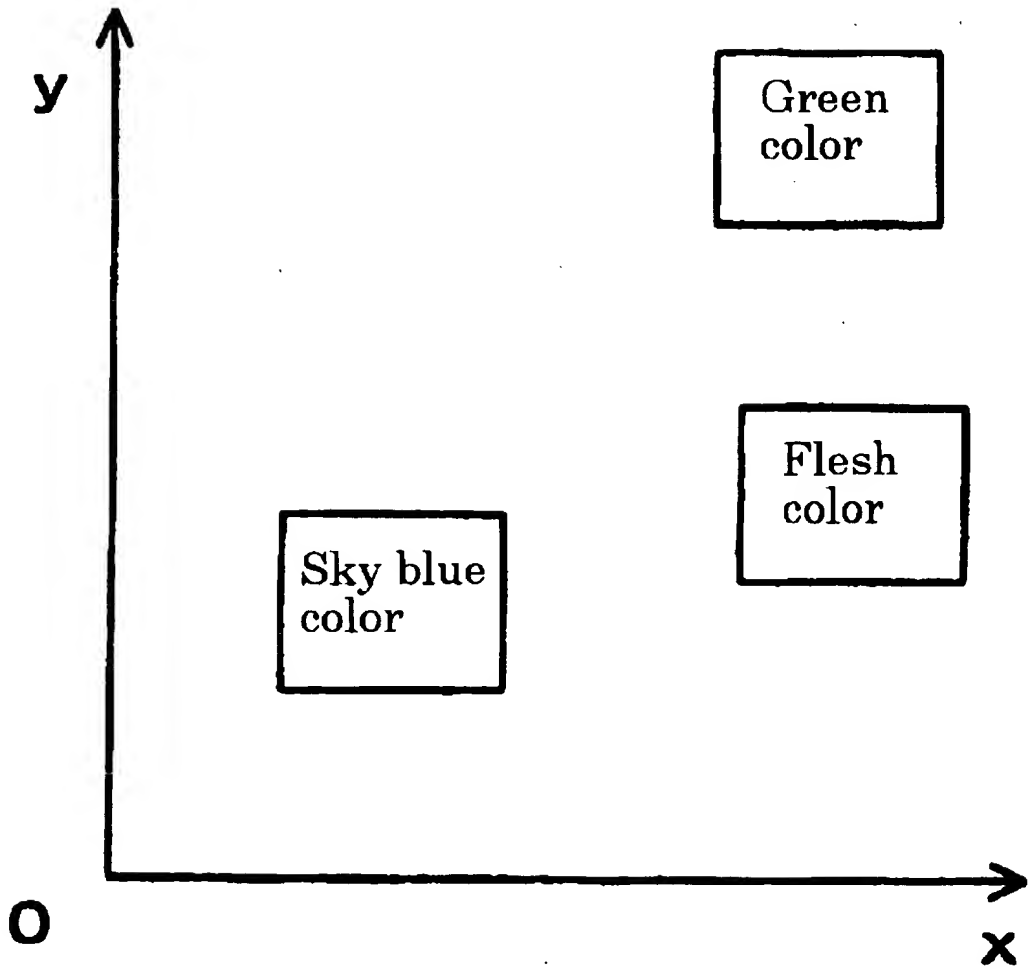


Fig. 15

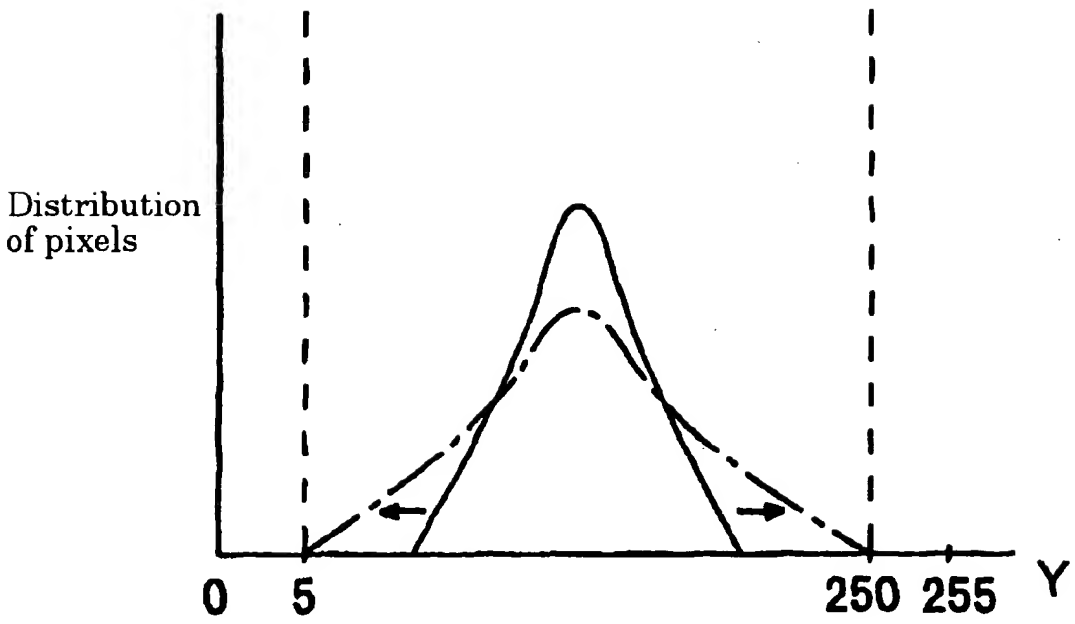


Fig. 17

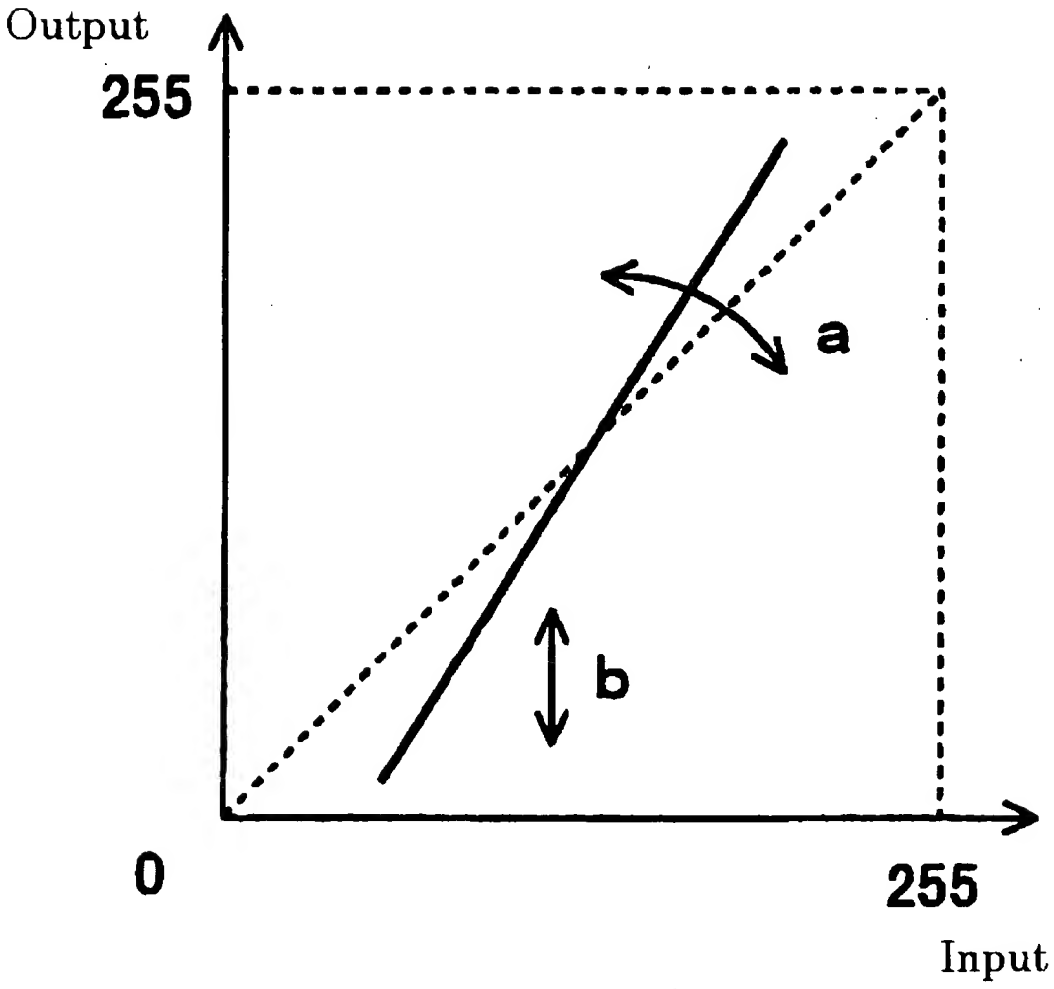
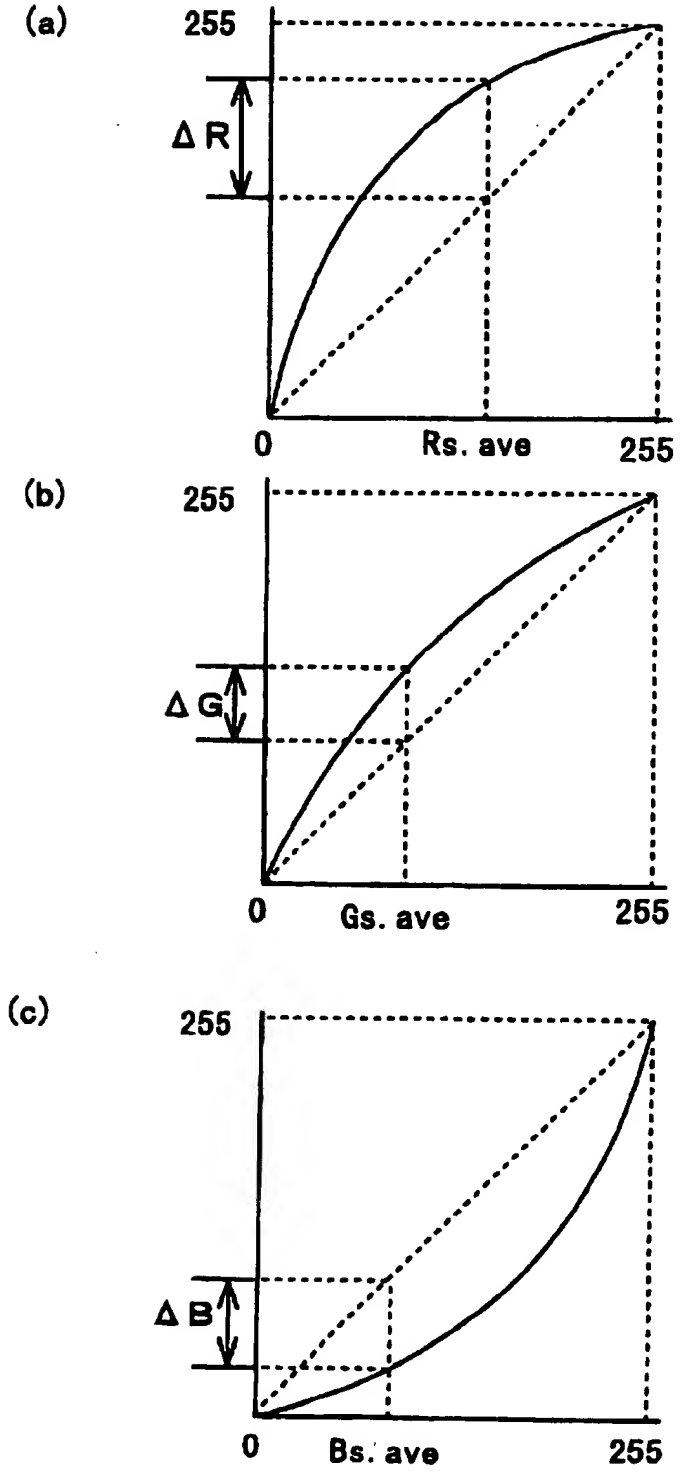


Fig. 27



the specified region, applicability k is changed in a range of '0' to '1'. More specifically, if the object is judged to be within the transition region at step S113, applicability k is adjusted toward '1' at step S114 in case the object is near the specified region or it is adjusted toward '0' in case the object is apart from the specified region. If the object is judged not to belong to the specified region nor the transition region, applicability k is set to '0' at step S115. While a center point of chromaticity is specified as exemplified in FIG. 12, there may be provided such an arrangement as shown in FIG. 14. Referring to FIG. 14, applicability k is set to '1' for an image processing object corresponding to radius 'r0', and under condition that a transition region is provided in a range of radius 'r0' to 'r1', applicability k is gradually adjusted toward '0' in a circumferential region.

At step S110, an applicable correspondence relationship is judged, and at steps S111 to S115, applicability k is determined as mentioned above. According to the present preferred embodiment, there is provided the unit for judging a correspondence relationship using software processing including such procedural steps as steps S110 to S115 mentioned above and hardware for implementation thereof. While applicability k is determined as a result of judgment in each image processing in the present preferred embodiment, it will be obvious to those skilled in the art that an alternative-choice judgment is also feasible in application.

At step S120, image processing is carried out for each object pixel according to applicability k. The following describes more specific conditions of image processing in practicing the present invention.

Contrast indicates a width of luminance in an entire image, and inmost cases of contrast adjustment, it is desired to increase a width of contrast. Referring to FIG. 15, there is shown a histogram of statistical luminance distribution of pixels of a certain image. In case of a solid-line curve of narrow distribution, a difference in luminance among bright and dark pixels is relatively small. In case of a dot-dashed-line curve of broad distribution, a difference in luminance among bright and dark pixels is relatively large, which signifies that a width of contrast is increased. Referring to FIG. 17, there is shown a graph indicating a luminance conversion operation for enhancing contrast. Assuming that the following equation holds for a relationship between original unconverted luminance y and converted luminance Y,

$$Y=ay+b \tag{9}$$

Through conversion under condition "a>1", a difference between a maximum value of luminance 'y max' and a minimum value of luminance 'y min' is increased, resulting in a broad distribution of luminance as shown in FIG. 15. In this case, it is preferred to determine slope 'a' and offset 'b' according to luminance distribution. For example, the following equations are given:

$$a=255/(y\ \max-y\ \min) \tag{10}$$

$$b=-a\,y\ \min \text{ or } 255-a\,y\ \max \tag{11}$$

Under the conditions indicated above, a certain narrow luminance distribution can be expanded to a reproducible extent. However, if it is expanded to an extreme end of a reproducible range, highlights may be blown out to white or shadows may be plugged up to black. To prevent this, a non-expandable margin corresponding to a luminance value of approx. '5' is provided at each of the upper and lower limits of the reproducible range. Resultantly, conversion parameters are expressed as follows:

$$a=245/(y\ \max-y\ \min) \tag{12}$$

$$b=5-a\,y\ \min \text{ or } 250-a\,y\ \max \tag{13}$$

In this case, conversion is not performed in ranges where "y<5" and "y>250".

In image data conversion, it is not required to perform calculation each time. If a luminance value range is '0' to '255', a result of conversion is predetermined for each luminance value and a conversion table is prepared as shown in FIG. 16. In this case, the conversion table is usable just for luminance. In an instance where image data contains direct elements of luminance, the conversion table can be used. Contrarily, in an instance where image data contains only indirect elements of luminance, the conversion table cannot be used. In most computer systems, red, green and blue elements of image data are indicated in gradation of brightness, i.e., color gradation data (R, G, B) is used. Since such gradation data (R, G, B) does not provide direct values of luminance, it is necessary to perform conversion to Luv color space scheme for determining luminance. This method is, however, disadvantageous since a large amount of calculation is required. Therefore, the following conversion expression is used, which is commonly adopted in television signal processing for directly determining luminance from RGB data:

$$y=0.30R+0.59G+0.11B \tag{14}$$

On a principle that linear conversion can be made between gradation data and luminance y as indicated above, Equation (9) is applicable to a relationship between original unconverted gradation data (R0, G0, B0) and converted gradation data (R1, G1, B1). Then, the following expressions are given:

$$R1=aR0+b \tag{15}$$

$$G1=aG0+b \tag{16}$$

$$B1=aB0+b \tag{17}$$

Consequently, the conversion table shown in FIG. 16 is applicable to gradation data conversion.

The following describes an image processing technique for adjusting brightness. As in the above-mentioned case of contrast adjustment, a histogram of brightness distribution is assumed. Referring to FIG. 18, a solid-line curve indicates a luminance distribution which has its peak inclined toward a dark level. In this case, the peak of entire distribution is shifted toward a bright side as indicated by a broken-line curve. Referring to FIG. 19, a solid-line curve indicates a luminance distribution which has its peak inclined toward a bright side. In this case, the peak of entire distribution is shifted toward a dark side as indicated by a broken-line curve. In these cases, linear luminance conversion as shown in FIG. 17 is not performed, but γ-curve luminance conversion is performed as shown in FIG. 20.

In γ-curve correction, entire brightness is increased when "γ<1", and it is decreased when "γ>1". A degree of this correction can be adjusted gradually by clicking an up-arrow or down-arrow of the BRIGHTNESS adjustment item on the processing menu area 43 shown in FIG. 7 as many time as required at step S100.

As in contrast adjustment, it is also possible to set up a value of γ automatically. As a result of our various experiments, it has been found that the following approach is advantageous: In a luminance distribution, median 'y med' is predetermined. If it is less than '85', an image of

interest is judged to be too dark and γ correction is made according to a γ value indicated below.

$$\gamma = y\ \text{med}/85 \tag{18}$$

or,

$$\gamma = (y\ \text{med}/85)^{**}(1/2) \tag{19}$$

Note, however, that even if "γ<0.7", a value of γ is set to 0.7 forcedly. Unless this kind of limit is provided, a night-scene image appears to be of daytime. If an image is brightened excessively, it becomes whitish entirely, resulting in low contrast. In this case, it is preferred to perform such a processing operation as enhancement of saturation in combination.

If median 'y med' is larger than '128', an image of interest is judged to be too bright and γ correction is made according to a γ value indicated below.

$$\gamma = y\ \text{med}/128 \tag{20}$$

or,

$$\gamma = (y\ \text{med}/128)^{**}(1/2) \tag{21}$$

In this case, a limit is also provided so that a value of γ is set to 1.3 forcedly even if "γ>1.3" for the purpose of preventing the image of interest from becoming too dark.

For this γ correction, it is preferred to provide such a conversion table as shown in FIG. **16**.

In edge enhancement processing for adjusting sharpness of an image, with respect to original non-enhanced luminance Y of each pixel, enhanced luminance Y' is calculated as expressed below.

$$Y' = Y + \text{Eenhance} \cdot (Y - \text{Yunsharp}) \tag{22}$$

where 'Eenhance' indicates a degree of edge enhancement, and 'Yunsharp' indicates unsharp-mask processing for each pixel of image data. The following describes the unsharp-mask processing: Referring to FIG. **21**, there is shown an example of an unsharp mask **60** comprising 5×5 pixels. The unsharp mask **60** is used in summation in such a manner that a center value of '100' is assigned as a weight to a processing object pixel 'Y(x,y)' in dot-matrix image data and a weight corresponding to a value in each array box of the unsharp mask **60** is assigned to each circumferential pixel thereof. In use of the unsharp mask **60**, summation is performed based on the following expression:

$$\text{Yunsharp}(x, y) = (1/396)\sum_{ij}(Mij \times Y(x+i, Y+j)) \tag{23}$$

In Equation (23), '396' indicates a total value of weighting factors. For an unsharp mask having a different size, a total of values in array boxes thereof is indicated. 'Mij' represents a weighting factor indicated in each array box of an unsharp mask, and 'Y(x,y)' represents each pixel of image data. In the unsharp mask **60**, 'i' and 'j' indicate coordinates on horizontal and vertical axes thereof.

Edge enhancement calculation based on Equation (22) provides the following functional meaning: 'Yunsharp(x,y)' is the result of addition in which a weight assigned to each circumferential pixel is reduced with respect to a pixel of interest, and accordingly an image is unsharpened through processing. Such an unsharpening operation is functionally equivalent to low-pass filtering. Therefore, "Y(x,y)−Yunsharp(x,y)" signifies that a low-frequency component is subtracted from each of original components, i.e., it is functionally equivalent to high-pass filtering. If a high-frequency component subjected to high-pass filtering is multiplied by a degree of edge enhancement 'Eenhance' and a result value of this multiplication is added to "Y(x,y)", it means that a high-frequency component is increased in proportion to the degree of edge enhancement 'Eenhance'. Thus, edge enhancement is accomplished. Since edge enhancement is required only for an edge part of an image, it is possible to reduce the amount of processing substantially by performing only in case that there is a large difference between adjacent pixels of image data.

In this case, the degree of edge enhancement 'Eenhance' can be adjusted by clicking an up-arrow or down-arrow of the SHARPNESS adjustment item on the processing menu area **43** shown in FIG. **7** as many times as required at step **S100**. Still more, it is possible to set up the degree of edge enhancement 'Eenhance' automatically.

At an edge part of an image, a difference in gradation data increases between adjacent pixels. This difference represents a gradient of luminance, which is referred to as a degree of edging. In an image, a degree of variation in luminance can be calculated by determining each of horizontal-direction and vertical-direction vector components. Although an object pixel in an image containing dot-matrix pixels is adjacent to eight pixels, a degree of variation is determined only with respect to adjacent pixels in horizontal and vertical directions for the purpose of simplifying calculation. Summation is performed on length values of respective vectors to represent a degree of edging 'g' for the object pixel of interest, and a sum value of edging degree is divided by the number of pixels to attain an average value. Assuming that the number of pixels is indicated as 'E(I)pix', a degree of sharpness 'SL' of an object image can be calculated as expressed below.

$$SL = \Sigma|g|/E(I)\text{pix} \tag{24}$$

In this case, as a value of 'SL' decreases, the degree of sharpness becomes lower (blurring). As a value of 'SL' increases, the degree of sharpness becomes higher (clearer imaging).

Since sharpness of an image depends on a visual sensation of an individual person, a degree of sharpness 'SL' is determined similarly using image data which has optimum sharpness attained experimentally. A value thus determined is set as an ideal level of sharpness 'SLopt', and a degree of edge enhancement 'Eenhance' is determined as expressed below.

$$\text{Eenhance} = ks \cdot (SL\text{opt} - SL)^{**}(1/2) \tag{25}$$

where coefficient 'ks' varies with a size of image.

In case that image data contains 'height' dots and 'width' dots in vertical and horizontal directions respectively, coefficient 'ks' can be determined as indicated below.

$$ks = \min(\text{height}, \text{width})/A \tag{26}$$

where 'min (height, width)' indicates the number of 'height' dots or the number of 'width' dots, whichever is smaller, and 'A' is a constant value of '768'. It is to be understood that the above value has been attained from experimental results and may be altered as required. Basically, as an image size increases, it is advisable to increase the degree of edge enhancement.

In the above-mentioned fashion, edge enhancement processing can be carried out in manual or automatic setting.

The following describes an image processing technique for adjusting saturation. In case of saturation adjustment

on the computer **21**. Note that a flowchart exemplified in FIG. **26** is for color adjustment processing to provide clear flesh color.

In the color adjustment processing, statistical calculation is performed on flesh-color-like pixels according to chromaticity of each pixel. As shown in FIG. **6**, each object pixel is moved for statistical calculation on all the pixels.

First, at step **S210**, chromaticity "x-y" of each pixel is calculated. As in the case of the example in the foregoing description, flesh color is identified if the following expressions are satisfied.

$$0.35<x<0.40 \quad (5)$$

$$0.33<y<0.36 \quad (6)$$

At step **S220**, it is judged whether or not chromaticity "x-y" converted according to each pixel of RGB gradation data is in a predefined flesh color range. If it is in the flesh color range, statistical calculation is performed on each pixel of color image data at step **S230**. This statistical calculation signifies simple addition of RGB gradation data values. The number of pixels is also counted to determine an average value for pixels judged to have flesh color, which will be described in detail later.

Thereafter, regardless of whether or not each object pixel is judged to have flesh color, each object pixel is moved at step **S240**. Thus, the above-mentioned sequence is repeated until it is judged at step **S250** that processing for all the pixels is completed. On completion of processing for all the pixels, step **S260** is performed to divide statistical result data by the number of pixels for determining an average value (Rs.ave, Gs.ave, Bs.ave).

Software processing for "x-y" chromaticity calculation at step **S210** and hardware for execution thereof provide the unit for judging chromaticity. It is judged at step **S220** whether or not chromaticity "x-y" is in a predetermined object range, and if the predetermined object range is satisfied, statistical calculation is performed on color image data at step **S230**. Then, at steps **S240** and **S250**, each object pixel is moved until all the pixels are taken. At step **S260**, statistical result is divided by the number of pixels to determine an average value. These software processing operations and hardware for execution thereof provide the unit for statistical calculation of object chromaticity pixels.

As to pixels having preferable flesh color, an ideal value (Rs. ideal, Gs.ideal, Bs. ideal) is predefined. In terms of memory color in psychology, each ideal value is different from result of actual measurement. In an example of flesh color, a person tends to have an optical illusion that slightly deviated flesh color is true rather than flesh color conforming to actual measurement result. This optical illusion is based on a stereotyped recognition of flesh color on photographs and pictures, i.e., it is referred to as a memory color effect in psychology. In the present invention, an ideal value is predefined in consideration of such a memory color effect so that color adjustment is made to eliminate a difference from the ideal value. Therefore, the ideal value may be in a wide target range expected without being biased to actual color.

Regarding flesh color pixels, a difference between an average value (Rs.ave, Gs.ave, Bs.ave) of RGB gradation data and an ideal value (Rs.ideal, Gs.ideal, Bs.ideal) predefined for preferable flesh color represents a degree of deviation in color image data fundamentally.

However, it is not preferred to apply the difference as a degree of color adjustment intactly. For instance, if the same degree of color adjustment is applied to all the pixels, a flesh color part may become satisfactory but colors of pixels on any parts other than the flesh color part may be affected significantly.

In the present preferred embodiment, therefore, a ratio of the number of flesh color pixels to the total number of pixels (flesh color ratio) is determined at step **S270** for regulating a degree of color adjustment. Degrees of adjustment of primary colors ΔR, ΔG and ΔB are expressed as shown below.

$$\Delta R=ks\ (Rs.\text{ideal}-Rs.\text{ave}) \quad (55)$$

$$\Delta G=ks\ (Gs.\text{ideal}-Gs.\text{ave}) \quad (56)$$

$$\Delta B=ks\ (Bs.\text{ideal}-Bs.\text{ave}) \quad (57)$$

Based on these equations, a value of flesh color ratio 'ks' is attained as indicated below.

$$ks=(\text{Number of flesh color pixels}/\text{Total number of pixels})$$

A degree of color adjustment thus attained is not applied intactly to color image data adjustment. In the present preferred embodiment, a tone curve is prepared using the degree of color adjustment at step **S280**. FIG. **27** is a diagrammatic illustration showing tone curves prepared in the present embodiment.

A tone curve represents an input-output relationship where RGB gradation data is converted with a degree of enhancement regulated. In an example of 256 gradations ranging from levels '0' to '255', a spline curve is drawn with respect to three identified output value points corresponding to gradation level gradation level '255' and a certain medium gradation level therebetween. Assuming that medium gradation level '64' is taken and output values are '0', '64' and '255', there is a coincidence in an input-output relationship even if input values are '0', '64' and '255', resulting in a tone curve being straightened. However, if output value '64' is not provided for input value '64', a gentle curve as shown in FIG. **27** is drawn to set up an input-output relationship. In the present preferred embodiment, a control point corresponding to the average value (Rs.ave, Gs.ave, Bs.ave) of RGB gradation data is used as a medium gradation level, and respective degrees of color adjustment ΔR, ΔG and ΔB are reflected in formation of a tone curve. In this fashion, the control point is changed so that each ideal value (Rs.ideal, Gs.ideal, Bs.ideal) is met when the flesh color ratio 'ks' is '1'.

At step **S290**, element colors of color image data are converted for all the pixels again using a tone curve thus attained to accomplish color adjustment of the color image data.

Software processing at step **S270** where color adjustment is made according to a ratio of the number of object pixels to the total number of pixels while determining a difference between a statistical result value and an ideal value, software processing at step **S280** where a tone curve is formed according to a determined degree of color adjustment, and hardware for execution thereof provide the unit for judging a degree of color adjustment. Software processing at step **S290** where color image data is converted and hardware for execution thereof provide the unit for adjusting color.

Having described the present preferred embodiment as related to flesh color adjustment for the purpose of simplicity in explanation, it is to be understood that color adjustment is not limited to flesh color. In consideration of a memory color effect in psychology, it is often desired to attain more vivid green color of tree leaves and clearer blue color of sky in addition to clearer flesh color through color

adjustment processing. Referring to FIG. **28**, there is shown a modified embodiment in which an object of adjustment is selectable.

In the example shown in FIG. **28**, an object of color adjustment is selected first at step **S305**. In use of the computer **21**, a window shown in FIG. **29** is presented on the display monitor **32** so that a human operator can select an object of color adjustment. The window exemplified in FIG. **29** is provided with a flesh color adjustment item for clearer flesh color, a green color adjustment item for more vivid green color of tree leaves, and a blue color adjustment item for clearer sky blue, each of which has a check box for allowing individual selection. In this example, duplicate selection is also permitted. When the operator turns on a desired check box and click the 'OK' button, a flag for each object thus specified is set up to start a loop processing of steps **S310** to **S350**.

In the loop processing, while moving an object pixel, statistical calculation is performed on each pixel through determining chromaticity as in the foregoing description. At step **S310**, object pixel chromaticity "x-y" is determined using Equations (1) to (4). Then, at step **S315**, a flesh color adjustment flag which has been set up at step **S305** is referenced to judge whether or not the operator has selected the flesh color adjustment item. If it has been selected, statistical processing for flesh color pixels is performed. This statistical processing is carried out in the same manner as at steps **S220** and **S230** in the previous example. If chromaticity "x-y" determined at step **S310** is in a predefined possible chromaticity range corresponding to flesh color, statistical calculation is performed for each element color of RGB gradation data.

At step **S325**, a green color adjustment flag is referenced to judged whether or not the operator has selected the green color adjustment item in the same manner as for flesh color adjustment. If the green color adjustment item has been selected, object pixel chromaticity "x-y" is checked to judge whether or not it is in a predefined possible chromaticity range corresponding to green color of tree leaves. If it is in the corresponding predefined chromaticity range, statistical calculation is performed at step **S330**. This statistical calculation is carried out in an area different from that subjected to flesh color statistical calculation.

Then, at step **S335**, the blue color adjustment item is checked to form a judgment in the same manner, and at step **S340**, statistical calculation is performed on another area.

At step **S345**, each object pixel is moved, and the above-mentioned sequence is repeated until it is judged at step **S350** that processing for all the pixels is completed. In this modified embodiment, a plurality of objects may be selected for color adjustment. Even in such a situation, statistical calculation is performed at steps **S315** to **S340** if chromaticity of each object pixel is in a predefined chromaticity range. Therefore, these processing operations provide the unit for statistical calculation of object chromaticity pixels.

At steps **S355** to **S365** after completion of chromaticity statistical calculation on all the pixels, a degree of adjustment for each color is calculated according to result of statistical calculation. Unlike the previous example, a processing operation for determining an average value from statistical calculation result is performed simultaneously with calculation of a degree of each color adjustment in this modified embodiment, and it is possible to modify relevant calculation procedures as required. As described in the previous example, each adjustment of flesh color, green color and blue color is carried out in the same manner. That is, an average value is calculated according to statistical calculation result, a difference between it and an ideal value predefined for preferable color is determined, and multiplication by a flesh color ratio, green color ratio or blue color ratio is performed for regulating a degree of each color adjustment.

Upon completion of step **S365**, there are provided three kinds of degrees of color adjustment since degrees of flesh color adjustment, green color adjustment and blue color adjustment have been determined through respective statistical calculations. Therefore, in the modified embodiment, addition is performed to reflect results of these statistical calculations inclusively. More specifically, in a situation of wider application of processing objects, degrees of color adjustment for respective processing objects ΔR, ΔG and B are determined as expressed below.

$$\Delta R = \sum_i ki(Ri \cdot ideal - Ri \cdot ave) \tag{58}$$

$$\Delta G = \sum_i ki(Gi \cdot ideal - Gi \cdot ave) \tag{59}$$

$$\Delta B = \sum_i ki(Bi \cdot ideal - Bi \cdot ave) \tag{60}$$

where

$$\sum_i ki <= 1 \tag{61}$$

$i = 1$: flesh color

$i = 2$: green color

$i = 3$: blue color

In this case, it is assumed that no duplicate counting is made.

At step **S370**, a tone curve is formed according to each of degrees of color adjustment ΔR, ΔG and ΔG thus determined, as shown in FIG. **27**. In this case, control points are indicated by Σ ki·Ri.ave, Σ ki·Gi.ave, Σ ki·Bi.ave. Software processing operations at steps **S355** to **S370** provide the unit for judging degree of color adjustment. After formation of each tone curve, color image data is adjusted at step **S375**.

The above-mentioned color adjustment device may also be implemented as a printer driver. In most cases, a printer driver is not capable of temporarily storing data in an output process after processing of input data. Hence, there is a certain limitation in functionality for changing processing conditions according to each region divided as desired. However, by setting up degrees of color adjustment for a plurality of elements as shown in Equations (58) to (60), it is possible to carry out effective color adjustment even for the printer driver having such a functional limitation.

The following describes operations of a preferred embodiment arranged with a printer driver.

As in the previous exemplary embodiment, it is assumed that a photographic color image shown in FIG. **30** is read in using the scanner **11** and printed out using the printer **31**. First, under condition that the operating system **21***a* is run on the computer **21**, the color adjustment application **21***d* is launched to let the scanner **11** read in the photographic image. When the photographic image thus read in is taken into the color adjustment application **21***d* under control of the operating system **21***a*, an object pixel applicable to processing is set at an initial position. Then, at step **S210**, chromaticity "x-y" of each pixel is calculated using Equations (1) to (4). At step **S220**, it is judged whether or not each

of values 'x' and 'y' is in a predefined flesh color chromaticity range. If it is in the flesh color chromaticity range, statistical calculation is performed on each pixel of color image data for each element color at step S230. In the photographic image shown in FIG. 30, pixels of person's hands, legs or face can be judged to have flesh color. In this example, statistical calculation is performed on a few percent of all the pixels as flesh color pixels. Then, at step S240, each object pixel is moved. Thus, the above-mentioned sequence is repeated until it is judged at step S250 that processing for all the pixels is completed. After completion of processing for all the pixels, statistical result data is divided by the number of flesh color pixels to determine an average value at step S260. At step S270, a difference between an ideal value of flesh color and the average value of flesh color pixels is determined, and it is multiplied by a flesh color ratio that represents a ratio of the number of flesh color pixels to the total number of pixels. At step S280, a tone curve is formed accordingly. Then, at step S290, based on the tone curve, each element color of color image data is converted for color adjustment. Since a degree of color adjustment is set at a moderate level with respect to the ideal value of flesh color in consideration of the ratio of the number of flesh color pixels to the total number of pixels, preferable color can be attained through proper color adjustment.

In the example shown in FIG. 30, a difference between an average value of flesh color pixels attained in statistical calculation and an ideal value of flesh color is multiplied by a flesh color ratio indicating a few percent value for regulating a degree of color adjustment. According to the regulated degree of color adjustment, a tone curve is formed for accomplishing color adjustment.

Still more, if a flesh color adjustment item, green color adjustment item and blue color adjustment item are selected in adjustment object selection as exemplified before, chromaticity "x-y" is calculated for all the pixels at step S310. Then, at steps S315 to S340, individual statistical calculation is performed for each adjustment object. In the example shown in FIG. 30, on a flesh color part of a person's image, a green color part of tree leaves and a sky-blue part of a background, each chromaticity "x-y" is applicable to each object range for statistical calculation.

After completion of processing for all the pixels, a degree of color adjustment for each object is determined in consideration of an occupancy ratio of object pixels at steps S355 to S365. At step S370, a tone curve is formed through regulating each degree of color adjustment thus determined. Then, at step S375, color adjustment is carried out for all the pixels of color image data. In this manner, flesh color adjustment for attaining clearer flesh color, green color adjustment for attaining more vivid green color of tree leaves, and blue color adjustment for attaining clearer sky blue in background are carried out through regulation according to the occupancy ratio of object pixels of each color.

After color adjustment thus accomplished, a color image is displayed on the display monitor 32 through the display driver 21c, and then if the color image thus displayed is satisfactory, it is printed out onto the printer 31 through the printer driver 21b. More specifically, the printer driver 21b receives RGB gradation image data which has been subjected to color adjustment, performs resolution conversion as predetermined, and carries out rasterization according to a print head region of the printer 31. Then, the image data thus rasterized is subjected to RGB-to-CMYK color conversion, and thereafter, CMYK gradation image data is converted into binary image data for output onto the printer 31.

Through the above-mentioned processing, the photographic color image read in using the scanner 11 is automatically subjected to optimum color adjustment. Thereafter, it is displayed on the display monitor 32, and then printed out onto the printer 31.

As set forth hereinabove, on the computer serving as the nucleus of color adjustment, chromaticity "x-y" of each pixel is calculated at step S210, and statistical calculation is performed at steps S220 to S230 if a value of chromaticity thus calculated is in a chromaticity range predefined for each color. After completion of statistical calculation on all the pixels, an average value is determined at step S260, and a degree of each color adjustment is calculated while taking account of an occupancy ratio of object pixels of each color at step S270. In this fashion, accurate statistical calculation is performed on color pixels to be adjusted independently of brightness, and a degree of each color adjustment is regulated by taking account of the number of pixels of each color in terms of occupancy ratio, thereby making it possible to carry out optimum color adjustment processing without giving an adverse effect on colors of pixels surrounding object pixels.

The invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. The present embodiments are therefore to be considered in all respects as illustrative and not restrictive, the scope of the invention being indicated by the appended claims rather by the foregoing description and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced therein.

What is claimed is:

1. A color adjustment device for performing color separation of color image data for each predetermined element color and adjusting said color image data in enhancement to provide each desired color in result of color image output delivered for each element color by an image output device, comprising:

a chromaticity judging unit which determines a value of chromaticity of each pixel according to said color image data;

an object chromaticity pixel statistical calculation unit which performs statistical calculation on pixels having chromaticity values determined, by said chromaticity judging unit, to meet a predetermined range of chromaticity;

a color adjustment degree judging unit which determines a degree of color adjustment so as to eliminate a difference between a predetermined optimum value for pixels meeting said predefined range of chromaticity and a result value attained in said statistical calculation and for regulating said degree of color adjustment according to an occupancy ratio of pixels subjected to said statistical calculation to the total number of pixels; and

color adjusting unit which carries out color adjustment of said image data according to the regulated degree of color adjustment;

wherein, for each given pixel of the color image data:

said chromaticity judging unit determines a value of chromaticity of said given pixel,

when said judged value of chromaticity of said given pixel meets a predetermined range of chromaticity, the object chromaticity pixel statistical calculation unit performs a statistical calculation with respect to the given pixel;

after said statistical calculation, said color adjustment degree judging unit determines a degree of color

# On Face Detection in the Compressed Domain

Huitao Luo and Alexandros Eleftheriadis
Advent Group, Columbia University
{luoht, eleft}@ctr.columbia.edu

**Abstract**

*We propose a fast face detection algorithm that works directly on the compressed DCT domain. Unlike the previous DCT domain processing designs that are mainly based on skin-color detection, our algorithm analyzes both color and texture information contained in the DCT parameters, therefore could generate more reliable detection results. Our texture analysis is mainly based on statistical model training and detection. A number of fundamental problems, e.g., block quantization, preprocessing in the DCT domain, and feature vector selection and classification in the DCT domain, are discussed.*

*Key words: face detection, DCT, JPEG, MPEG.*

## 1 Introduction

Human face detection is an interesting research topic. In the literature, many works have been reported with different application backgrounds. They could be classified into two groups, *i.e.*, color based approaches and texture based approaches.

Color based approaches have been popular in multimedia community because of their relative simple design and fast performance. In general, this type of algorithms tries to model human skin color in different chromatic spaces (RGB, YCbCr, HSV, *etc.*) with various statistical models. Typical skin color modeling works include: *mixture of Gaussian* [13], *linear region approximation* [7], *Bayesian minimal cost decision rule* [18], *etc.* In addition to color modeling, a number of recent works seek to include additional heuristics such as texture, symmetry, region ratio, region segmentation and merging, *etc.* [2, 19, 1]. In order to further improve the processing speed, Wang and Chang [18] proposed to detect human faces directly on the compressed MPEG macroblocks. In their work, JPEG pictures and MPEG I frames are partially decoded (entropy decoded and de-quantized) to restore DCT parameters in their block structure. Their algorithm then works directly on the decoded DCT parameters. Color information is used as the major detection clues in their algorithm. A skin color model is created at the macroblock level in the YCbCr color space. In addition, they also use some texture information by grouping the DCT parameters into bins and evaluating the energy distribution patterns based on bin statistics. This algorithm has interesting fast performance, but it shares the same problem with those typical color-based algorithms designed in the pixel domain, *i.e.*, it has good detection rate, but suffer from high false alarm rates when the backgrounds have skin like colors.

In contrast, texture based works detect face directly from the picture grayscale information. Most texture based methods are developed from face recognition algorithms, because in the recognition sense, color information does not help much in distinguishing different individual's faces. Instead, it is the face texture that we use to recognize different persons. Typical examples [1] based on this observation includes Sung and Poggio [16]' system developed at MIT AI lab and Rowley and Kanade [15]'s system developed at CMU Robotic Institute. Both systems use similar preprocessing algorithms and multi-scale searching mechanism, except that Sung and Poggio's system uses multimodal Gaussian clustering method as classifier while Rowley and Kanade's system uses neural network as classifier. Similar systems available in the literature also include: Lew [9]'s work that uses information theory, Collobert *et al.* [5]'s and McKenna [10]'s works that use neural networks, Yang and Ahuja [20]'s work that uses factor analyzer and Osuna *et al.* [14]'s work that uses support vector machine (SVM) as classifiers. They all reported close detection performance on some common testing pictures, for example the CMU testing database. In general, texture-based algorithms are more reliable than color based algorithms, but they are also more complex and slower.

In this paper, we propose a face detection algorithm that combines both color and texture information in order to find a good balance between speed and detection reliability. We design the algorithm to work on the com-

[1] We do not intend to give a comprehensive survey of face detection algorithms. Instead only a group of similar algorithms that our work bases on is covered.

pressed DCT domain in a similar way as Wang [18]'s work. However, our work extends theirs by building up an independent texture-based detection module in the compressed DCT domain, *i.e.*, we study how to map the successful texture-based face pattern detection algorithms such as [16, 15, 20, 14] from the traditional pixel domain to the DCT transform domain. With the new texture-based detection model included, face detection problems can be solved more reliably in the compressed domain, as compared with Wang and Chang [18]'s work. We believe this algorithm is especially valuable for fast content analysis of large amount of visual media data stored in the compressed formats such as JPEG and MPEG. In the following of this paper, when mentioning compressed DCT domain, we refer to JPEG pictures and MPEG I frames that are partially decoded (entropy decoded and de-quantized) and have their DCT parameters restored in 8 pixel by 8 pixel block structures, if not expressed clearly otherwise.

The structure of this paper is as follows. First in Section 2, we give a simple survey of the texture-based face detection design in the pixel domain. In Section 3 we discuss in detail the texture-based face detection algorithm design in the compressed DCT domain. In Section 4, the texture-based face model is extended to include color information and a new combined texture-based and color-based face detection algorithm is developed with experimental results. Finally in Section 5 we conclude the paper.

## 2 Texture Based Face Detection in the Pixel Domain

In this work, we seek to map the successful texture-based face detection algorithms from the original pixel domain to the transformed DCT domain. Therefore, before going directly to the DCT domain, we first take a close look at the available works in the pixel domain.

In the available face detection works designed on the pixel domain, successful algorithms such as [16, 15, 20, 14] generally share similar processing procedures and structures. In these works, the face pattern is represented as a rectangle or square window of pixel plane. In order to detect face patterns, the systems scan the input image plane at every location. That is, the image is divided into multiple (possibly overlapping) subimages of the model window size. Each window is compared with a previously trained "face" model to tell whether it is a face pattern or not. In order to detect faces in multiple scales, the input image is downscaled to a series of scales (for example, by scales of $1.2^n, n = 1, 2, \ldots$) and the detection process is repetitively applied to each scale. Or in other words, to tell a windowed image a face or not, the window is always first scaled to the size of the model, and then compared with it.

The details of this processing flow chart are illustrated in Fig. 1. As we can see from Fig. 1, to detect faces, the input image is scaled and cropped to generate a windowed image pattern. The pattern is processed with a preprocessing module to remove illumination noise and normalize the grayscale ranges. The normalized image pattern is then fed to a classifier to see if it is a face pattern. To create a face model, a database of frontal faces is generally used. Each face is scaled and moved to align its common facial feature points such as

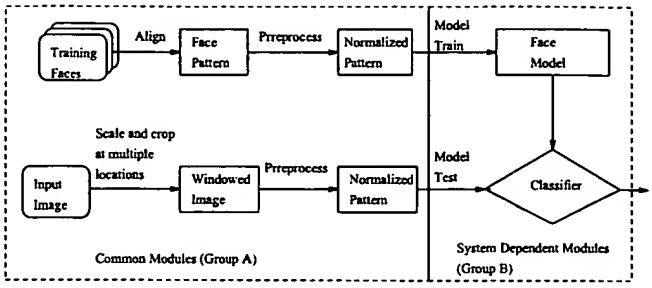



Figure 1: Illustration of common face detection procedures in the pixel domain

corner of eyes, tip of nose, *etc.* to specified positions in the model window. The windowed image pattern is then processed by a preprocessor, and finally applied to train a classifier.

In all the four detection systems [16, 15, 20, 14], their basic structures are generally the same (as indicated in modules covered by the dashed rectangle in the left (Group A) of Fig. 1). The only different part is the classifier, *i.e.*, classifier's internal structure, its training and detection operation (as indicated in modules covered by the dashed rectangles in the right (Group B) of Fig. 1.

Based on the summery as depicted in Fig. 1, the detection complexity is $M$ times of the processing in Group A plus the processing in Group B. $M$ depends on the image size, the model size, and the range of the face sizes that are to be detected by the system.

However, the essence of the complexity issue is the size of the face model, which determines directly the complexity of the classifier, *i.e.*, how complex the classifier should be, in order to separate the *face patterns* from the *nonface patterns*. Though there is no clear measure available to judge the complexity of face pattern classification, most reported papers use similar face model sizes. For example, Sung [16] uses a (masked) 19-pixel by 19-pixel square window, Rowley [15] uses a (masked) 20-pixel by 20-pixel square window, and Collobert *et al.* [5] uses a 15-pixel by 25-pixel rectangle window. Therefore, the face detection problem is (at most)[2] a 2-dimensional pattern classification problem at the size of about 20 pixels by 20 pixels. If we stack up the pixels row by row as in [16], the problem is then converted to a 1-dimensional pattern classification problem at the size of about 200-400 dimension.

## 3 Texture Based Face Detection in the DCT domain

### 3.1 Feature Representation in the Block DCT Domain

#### 3.1.1 The DCT Transform

In principle, pattern detection models should not be influenced by converting the problem to the DCT domain

[2]Sung [16]'s work indicates that the classification problem can be projected to subspaces in much lower dimensions. But there is no clear quantitative boundary on how low this dimension could be.

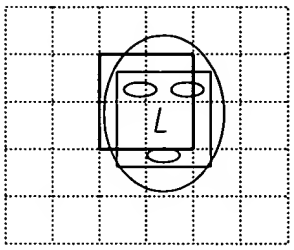


Figure 2: Illustration of the block quantization problem in face detection.

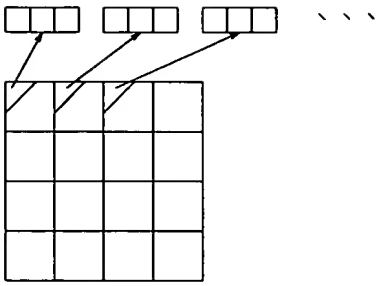


Figure 3: Illustration of the feature vector creation from DCT parameters

if we do not consider the blocks and quantization errors incurred. Because DCT is an orthonormal transform, both the *Euclid distance* and the *Mahalanobis distance* are unchanged after the transform. If we follow the Gaussian clustering approach as discussed in [16], it is easy to proof that the Gaussian model remains a Gaussian model in the DCT domain. In addition to the invariant features, the DCT domain is better than the pixel domain for pattern classification problems in that DCT transform reduces the dependence among individual components and compresses the feature energy to the low frequency parameters. Therefore, it is much easier to choose feature components from DCT parameters than from direct pixel values.

#### 3.1.2 Block Quantization Problem

However, to apply model-based algorithms directly to the compressed domain of JPEG and MPEG, a major problem to overcome is that the image frames are divided into 8 by 8 blocks before DCT transform. Therefore, any detection work based on DCT parameters has to be done at the locations of blocks rather than pixels. That is, the blocks reduce the spatial resolution of the system by 8, which makes it hard to detect small faces without fully decoding the image from the block based DCT parameters to pixels. We refer to this problem as *block quantization* in this work.

More specifically, the block quantization problem influences face detection in the following three aspects.

First, because the DCT parameters are organized in a block structure, in order to detect a face, a face model cannot be used to search the picture pixel by pixel. Instead, this search can only done block by block in the DCT domain. This problem is better illustrated in Fig 2, in which the rectangle in light solid lines represents the actual location of a face, while the rectangle in dark solid lines is the closest searching window that the system can arrive at based on 8 by 8 block quantization. Therefore, in order to detect faces that are not aligned with block positions, we need to introduce certain translation invariant feature in face model training. Or in order words, when training the face model, more images patterns should be included as positive training examples than the corresponding procedure that designed in the pixel domain.

Second, in addition to the feature aligning problem, block quantization also introduces some background noise when the searching window is not well aligned with the actual face region, which influences the accuracy of both model training and detection.

Third, in the block-based DCT domain, it is hard to obtain resolution transformation. Though several papers [8, 11, 4] have discussed the issue of fast resolution transform in the compressed DCT domain, it is still too expensive to carry out resolution transforms in arbitrary ratios, *e.g.*, to down-sample the image by 1.2. Therefore, in order to detection faces in multiple scales, we have to design models individually for each scale. We call this solution as a *multi-model* approach in this work, as compared to the multi-scale approach commonly used in the original pixel domain.

To sum up, the block quantization reduces the distribution density of the face patterns in the feature space, as well as introduces background noises and the scaling problem. Therefore, to detect human face patterns within the block based DCT domain is more difficult than to do it in the pixel domain.

#### 3.1.3 Feature Vector Design

In this work, we design face detection as a 1-dimensional vector classification problem similar to Sung [16]'s system. In the DCT domain, feature vectors are created directly from (block based) DCT parameters as follows. Suppose the size of the face model is $M$ block by $M$ block and the desired length of the feature vector is $N$, then in each DCT block, the lowest $d$ DCT parameters are used for feature vector creation, where

$$d = N/(M * M).$$

This is better illustrated in Fig. 3, in which the lowest d DCT parameters from each block is stacked up to form a $N$ dimension feature vector.

In order to choose the first few low frequencies from a 2-D DCT block, we number the 64 DCT parameters according to the typical DCT quantization table design as reported in [17], *i.e.*, the larger the quantizer is, the less important its corresponding DCT parameter is for low frequency representation of the picture. In Eq.1, we show the positions of the first 16 parameters.

$$\begin{pmatrix} 1 & 2 & 3 & 10 \\ 4 & 5 & 7 & 13 \\ 8 & 6 & 11 & 15 \\ 9 & 12 & 14 & 16 \end{pmatrix}. \quad (1)$$

Based on the complexity analysis in Section 2, the complexity of face detection problem should be pro-

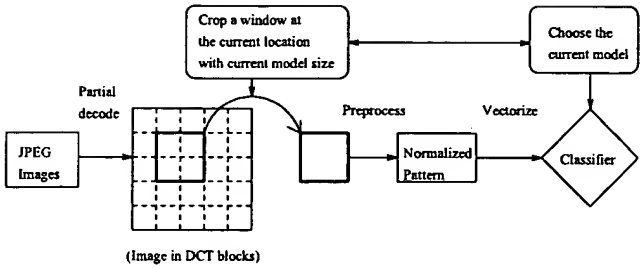



Figure 4: Illustration of the face detection procedures in the DCT domain.

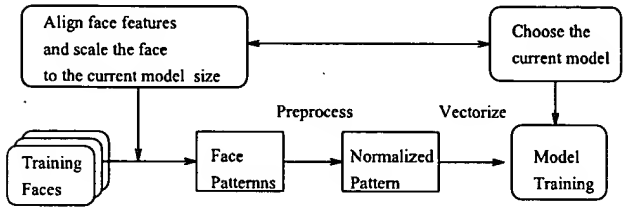



Figure 5: Illustration of the face model training procedures in the DCT domain.

cessed adequately over a rectangle window of 19 pixel by 19 pixel, therefore, the feature vector length in this work should be in the range of 200-400 dimension. For example, if a 5 block by 5 block face model is to be created, and we choose the feature vector length to be 300, then for each DCT block, its lowest 12 DCT parameters are used to create the feature vector for training and classification.

In principle, this approach of choosing the lowest few DCT parameters in each block is actually to downscale the windowed image to a *unit* model size, saying 19 pixel by 19 pixel, as used in the pixel domain. In other words, to tell a windowed image pattern a face or not, in the pixel domain, the face is downscaled to the unit model size, *e.g.*, 19 by 19 pixel, and compared with the face model. In the DCT domain, the correspondence is that the low frequent DCT parameters in each block of the windowed image pattern are used to form a feature vector, which is then compared with the face model. In this mapping, as long as the lowest few DCT parameters have maintained the image features at the resolution of 19 by 19 pixels (approximately)[3], we have reason to believe that those parameters have maintain the necessary information for face detection, based on our complexity analysis in Section 2. The benefit in this domain mapping is that unlike the bilinear downsampling in the pixel domain, choosing the lowest few DCT parameters is an easier and better way to do downsampling in the DCT domain. This is also noticed and discussed by Dugad and Ahuja [4] in their paper on fast DCT domain downsampling design.

[3]That is, if the image is decoded with only these low frequent parameters and then downscaled to 19 by 19 pixels, the image quality is still comparable with direct downloading the original image in the pixel domain.

### 3.2 Face Detection System Design

Based on the analysis in Section 2, a multi-model DCT domain face detection system is designed that combines algorithm capabilities and performance efficiencies.

#### 3.2.1 Multi-Model Detection System

Due to the block quantization problem as discussed in Section 3.1.2, model-based face detection in the block-based DCT domain faces two problems, *i.e.*, aligning problem and scaling problem.

For the aligning problem, because the block size is 8 pixels, there are totally 64 possible spatial setups at the every searching position in the DCT domain. To solve the aligning problem, one might train 64 models for one model size, with each one representing a face pattern at a different aligning position. However, this approach is obviously not efficient because 64 models are hard to store as well as to apply. In addition, there are redundancies in the 64 models as the spatially neighboring models are similar to each other. Therefore, the trade-off has to be made between model efficiency and model accuracy. In addition, to overcome the scaling problem in the DCT domain, multiple models have to be created in order to detect faces in different scales, which further improves the burden of choosing too many models in one scale.

In this system, we create face models in six scales, and the model windows are designed to be squares in side length of 5, 6, 7, 8, 9, and 10 DCT blocks. That is, faces in the range of 40 by 40 pixels to 80 by 80 pixels are covered by the system. For each scale, one face model is trained for faces in all the possible aligning positions, *i.e.* the model represents variations in different face features as well as in different aligning positions (all the 64 possible positions).

The processing procedures for each model size is generally the same as those in the pixel domain (refer to Fig. 1), except that the works are moved to the DCT domain. We illustrate the DCT domain detection procedures and model training procedures separately in Fig. 4 and Fig. 5. The details of the processing modules are discussed in the following sections.

#### 3.2.2 Preprocessing and Masking in the DCT Domain

To remove the signal variance introduced by different illuminations and different grayscale dynamic range, a number of preprocessing steps are used in the works designed in the pixel domain. In the DCT domain, we find it also possible to implement their correspondences. More specifically, our system includes the following preprocessing steps:

1. **Masking.** Similar to Sung [16]'s work, we introduce binary face masks on top of DCT blocks. For face patterns, these masked blocks often represent background regions. Removing them from the feature vectors ensures that the subsequent modeling work does not wrongly encode any unwanted background structures. Based on different sizes of the face model, different masks are designed to represent different spatial resolutions. In Fig. 7, we show the six masks we used in our system for face

models in six different scales. Note that the face models are actually in different sizes. They are scaled to the same size in Fig. 7 for ease of illustration. Each block in the model windows is of size 8 pixel by 8 pixel.

2. **Illumination Linear Factor Correction.** In order to remove shadowing effect, a 2-dimensional linear function is fitted to the DC plane of the face region in the DCT domain. The fitted function is then removed from the DC plane.

3. **Histogram Equalization.** This nonlinear process is applied directly to the DC components of the face region DCT blocks. In addition, the AC components in each block are changed linearly to reduce the block effects. That is, for each DCT block, if its DC component $d$ is mapped to $d'$ according to the histogram equalization, then its AC components $a_i$ are also mapped linearly to $a_i'$ with $a_i' = a_i * d'/d$.

Fig. 6 shows an example of preprocessing (the picture is taken from Olivetti face database[4]). Fig. 6(a) is a cropped face region, (b) is the face region with a grayscale linear factor removed, and (c) is the final result of face region preprocessing. We can see that the shading in the original face is effectively removed (in subfigure (b)). In addition, the histogram equalization based on DC components introduces certain blocking effect, but the general quality is acceptable (in subfigure (c)).


Figure 6: An example of preprocessing results on block based DCT domain

### 3.2.3 Distribution Based Classifier Design

For face detection purpose, a number of classifiers have been used. These include: unimodel Gaussian [12], multimodel Gaussian [16], neural networks [15, 10] and support vector machine [14]. Among them, neural networks and support vector machine have been shown to have theoretical merits for high dimensional vector classification problems. Especially SVM is proved capable of finding optimal classification boundary in that it can minimize structural risk [3]. However, both neural networks and SVM are hard to train, *i.e.*, to organize the positive and negative training samples in order to make the classifier converge to the optimal status.

In this work, we have to train multiple face models for faces in multiple scales, therefore, we choose to use multimodal Gaussian model as the classifier, which is a trade-off between model training complexity and classification performance.

[4]http://www.cam-orl.co.uk/facedatabase.html

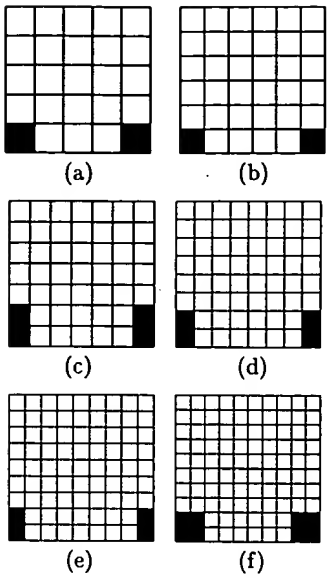



Figure 7: Illustration of binary face masks design for models in different sizes.

The essence of multimodal Gaussian model is to approximate the feature vector distribution with a number of Gaussian clusters. Though Gaussian distribution is a general purpose statistical model that has been extensively used, unimodel Gaussian distribution is shown by Sung [16] inadequate to model face patterns' distribution in the high dimensional feature space. Multimodal Gaussian model is a natural extension of unimodal Gaussian models that has worked well in complex and high dimensional distribution problems. It also has moderate training complexity as compared with neural networks and SVMs. In this work, in order to improve classification performance, we use two groups of multimodal Gaussian models to approximate separately the distribution of face patterns and nonface patterns.

**Positive Training Samples** The basic idea of this approach is to approximate the distribution of face feature vectors with high dimensional Gaussian clusters. To study the distribution of face patterns in the DCT domain, 300 frontal faces are collected from various sources, such as MIT face database, Yale face database, Olivetti face database, University of Sterling face database[5] and some anchorperson images from the NBC news video database stored at AT&T research lab. Four feature points of each face, *i.e.*, the inner and outer corners of both eyes, are marked manually. Each face is moved and scaled to align these feature points with specific positions in a model window. The face image is then cropped with the model window, and converted to the DCT domain. After that, the feature vector is obtained

[5]Most of them are downloadable from http://www.cs.rug.nl/~peterkr/FACE/face.html

from the DCT parameters as described in Section 3.1.3. Because we train only one model for one modeling window size, feature vectors should also be created to represent face patterns whose feature points are not perfectly aligned with the face model. In this work, we use 16 out of 64 possible spatial aligning positions for each training face to generate training samples. In addition, to make use of the symmetric property, each training face is flipped and used as a new training sample. Therefore, in this system, totally $16*2*300 = 9600$ feature vectors are used as positive training samples.

**Clustering Algorithm** We cluster the face feature vectors into six clusters of Gaussian distribution. The clustering algorithm used here is similar to the K-means algorithm, except that the *Euclidean distance measure* is replaced by a *logarithmic Gaussian distance measure*. If we denote each cluster with a Gaussian distribution $N(\mathbf{v}_i, \mathbf{C}_i)$. A new feature vector's distance to each cluster is defined to be a *logarithmic Gaussian distance* as:

$$d = \frac{1}{2}(N \ln 2\pi + \ln|\mathbf{C}_i| + (\mathbf{v} - \mathbf{v}_i)^T \mathbf{C}_i^{-1}(\mathbf{v} - \mathbf{v}_i)),$$

where $N$ is the dimension of the feature vectors. Because $N$ is a rather high dimension in this problem ($200 \sim 400$), we actually use Karhunen-Loeve transform to reduce the above equation into a lower dimension problem as:

$$d = \frac{1}{2}\left[M \ln 2\pi + \sum_{k=0}^{M-1} \ln|\lambda_k| + \sum_{k=0}^{M-1} \frac{y_k^2}{\lambda_k}\right],$$

where $y_k^2$ are the principle components and $\lambda_k$ are the eigenvalues. $M$ is the number of eigenvectors used to approximate the system. Generally $M \ll N$.

The detailed clustering steps are as follows.

1. Initialize the clustering process by grouping the feature vectors into six groups in *Euclidean* distance space, *i.e.*, a vector is put into the group whose center is the closest to it among the six groups. And the covariance matrix for each cluster is initialized to be unit matrix.

2. Re-compute the data centers of each cluster to be the center of the current cluster partition.

3. Based on the current cluster centers and covariance matrixes, re-assign the data partition by assigning the feature vectors to the cluster that is closest to it in the *logarithmic Gaussian distance* space. If the difference between the new data partition and the old one is bigger than a threshold and the inner loop time (Step 2 and Step 3) is less than the maximal time, goto Step 2, otherwise goto Step 4.

4. Re-compute the covariance matrixes of the 6 clusters based on the current data partition.

5. Based on the current cluster centers and covariance matrixes, re-assign the data partition by assigning the feature vectors to the cluster that is closest to it in the *logarithmic Gaussian distance* space. If the difference between the new data partition and the old one is bigger than a threshold and the outer loop time (Step 2 to Step 5) is less than the maximal time, goto Step 2. Otherwise, return the created mean vector and covariance matrix of each cluster.

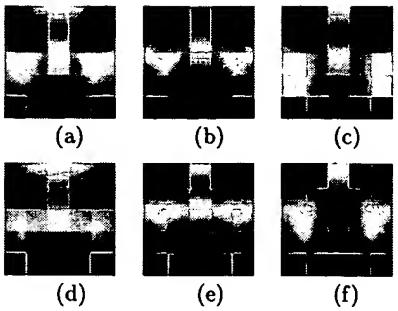



Figure 8: *Average faces* of the six face clusters when the face model size is 5 blocks by 5 blocks.

In Fig. 8, we give the clustering results on the 9600 face feature vectors. The model size used is 5 blocks by 5 blocks. Fig. 8(a) to (f) are the average faces of the six clusters at the time of convergence. Because only the 4 lowest frequent DCT parameters (the choice of 4 will be discussed later in this section) in each block are used for feature vector creation, the block effect is noticeable. But in general the six mean faces represent mainly the variances in face textures rather than those in different spatial aligning positions.

**Negative Training Samples** In our experiments, we find many face like "nonface" patterns in our testing examples that can not be simply separated from face patterns by thresholding. In order to reduce the misclassification rate, we further create a multimodal Gaussian model for these face like negative patterns.

The training samples are collected in a *boot-strap* fashion. That is, the positive face model is first created by the clustering algorithm as previously discussed. Based on this model, a face detector is designed, which is then applied to the training pictures. The nonface patterns misclassified as faces by the face detector are used as negative samples.

In this way, we collect about 9000 negative nonface samples and cluster them into eight clusters. The clustering algorithm used is the same as positive face sample clustering. The clustering result of model size 5 blocks by 5 blocks is shown in Fig. 9, in which each subfigure represents an average nonface pattern for the eight clusters.

**Classification** The classification problem is based on the distance measures from the input feature vector to the positive and the negative clusters. Let's denote the input vector as $\mathbf{v}$, the six positive clusters as $N(\mathbf{C}_k^{(p)}, \mathbf{v}_k^{(p)})$, $k = 1, 2, \cdots, 6$, and the eight negative clusters as $N(\mathbf{C}_k^{(n)}, \mathbf{v}_k^{(n)})$, $k = 1, 2, \cdots, 8$. Then a 6-dimension positive distance vector could be defined as

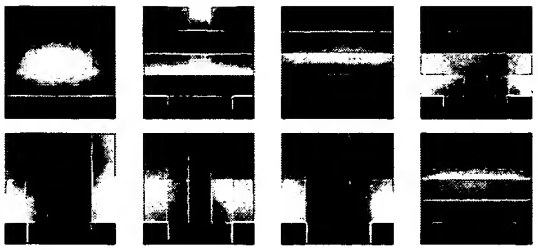


Figure 9: *Average nonfaces* of the eight nonface clusters when the face model size is 5 blocks by 5 blocks

$\mathbf{d}^{(p)} = (d_1^{(p)}, d_2^{(p)}, \cdots, d_6^{(p)})$, where

$$d_k^{(p)} = \frac{1}{2}(N \ln 2\pi + \ln|\mathbf{C}| + (\mathbf{v} - \mathbf{v}_k^{(p)})^T \mathbf{C}^{-1}(\mathbf{v} - \mathbf{v}_k^{(p)})), \quad (2)$$

where $N$ is the dimension of the feature vector $\mathbf{v}$. In practice, because the input feature vectors are in high dimension, the covariance matrix $\mathbf{C}$ is always decomposed with KL transform:

$$\mathbf{C} = \mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1},$$

where $\mathbf{T}$ is the eigen-matrix and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues. With only the first M eigenvectors of the eigen-matrix $\mathbf{T}$ used to span the feature space, the distance Eq. 2 is decomposed into two parts: the distance in the feature space (DIFS) and the distance from the feature space (DFFS),

$$\text{DIFS} = \frac{1}{2}\left[M \ln 2\pi + \sum_{k=0}^{M-1} \ln|\lambda_k| + \sum_{k=0}^{M-1} \frac{y_k^2}{\lambda_k}\right],$$

$$\text{DFFS} = \frac{1}{2}\left[(N-M)\ln 2\pi + (\ln|C| - \sum_{k=0}^{M-1} \ln|\lambda_k|) + \frac{\Delta^2}{\rho}\right],$$

where $y_k$ is the principle components and $\Delta^2 = ||\mathbf{v}|| - \sum_{k=0}^{M-1} y_k^2$ is the residue. $\rho$ is a weighting factor based on the estimation of eigenvalues $\lambda_k, (k = M, M+1, \cdots, N)$. In this work, the distance measure is therefore defined as:

$$d_k^{(p)} = \frac{1}{2}\left[N \ln 2\pi + \ln|\mathbf{C}| + \sum_{k=0}^{M-1} \ln|\lambda_k| + \eta\frac{\Delta^2}{\lambda_M}\right], \quad (3)$$

where $\eta$ is an adjustable weighting factor.

Similarly, an 8-dimension negative distance vector is defined as $\mathbf{d}^{(n)} = (d_1^{(n)}, d_2^{(n)}, \cdots, d_8^{(n)})$, with respect to the eight negative clusters.

Therefore, the classification problem is reduced from a high dimension problem ($N = 200 \sim 400$) to a lower dimension problem with $N = 14$, which is then solved with a simple minimal distance classification algorithm in our work, *i.e.*, if

$$\min_{k=(1,\cdots,6)} d_k^{(p)} \leq \min_{k=(1,\cdots,8)} d_k^{(n)},$$

the pattern is detected as a face, otherwise it is a nonface.

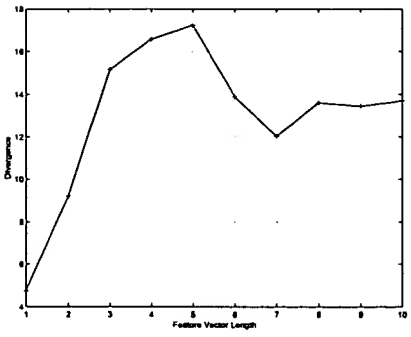



Figure 10: Illustration of the relation between the feature vector length and the divergence of the face and nonface patterns.

In implementing the classifier on practical face images, we tried to select feature vectors of a variety of lengths. Based on the complexity analysis in Section 2, the complexity of the face detection problem requires feature vectors of length in the range of 200-400 dimensions. However, we notice that when feature vectors coming from various spatial aligning positions are included as positive training samples (due to the block quantization problem), the high frequency DCT parameters become unstable in both face model training and face detection functions. In order to illustrate this problem, an experiment is carried out to measure the *separability* feature between the 9600 positive samples and the 9000 negative samples under the condition of different feature vector lengths.

The measure we used here is the *divergence* measure as defined in [6]. The divergence measure of two Gaussian distributions $N(\mathbf{v}_1, \mathbf{C}_1)$ and $N(\mathbf{v}_2, \mathbf{C}_2)$ is given as

$$\text{Div} = \frac{1}{2}(\mathbf{v}_1 - \mathbf{v}_2)^T(\mathbf{C}_1^{-1} + \mathbf{C}_2^{-1})(\mathbf{v}_1 - \mathbf{v}_2) + \frac{1}{2}\text{tr}[(\mathbf{C}_1 - \mathbf{C}_2)(\mathbf{C}_2^{-1} - \mathbf{C}_1^{-1})] \quad (4)$$

To study the influence of the feature vector length on the separability of the problem, both the 9600 positive samples and the 9000 negative samples are projected to the six positive clusters, which generates two groups of feature vectors in Gaussian distribution (in $\mathcal{R}^6$). Their divergence is then computed according to Eq. 4. In this experiment, we used a model size of 5 blocks by 5 blocks (40 by 40 pixels). The feature vector length is changed from 23 to 230, or 1 parameter per DCT block to 10 parameters per DCT block. The corresponding divergence is computed and their relation is illustrated in Fig. 10.

From Fig. 10, we notice that the divergence increases with the feature vector length from 1 to 5 (parameters per block), and then drops when the feature vector length further increases. That is, the more DCT parameters are included into the feature vector, the less likely that the face patterns are able to be separated from the nonface ones. This problem comes mainly from the high frequency components in each DCT blocks. When we try to build up one model to represent face patterns in all the 64 different spatial aligning positions (with respect to the current model window), the high frequent

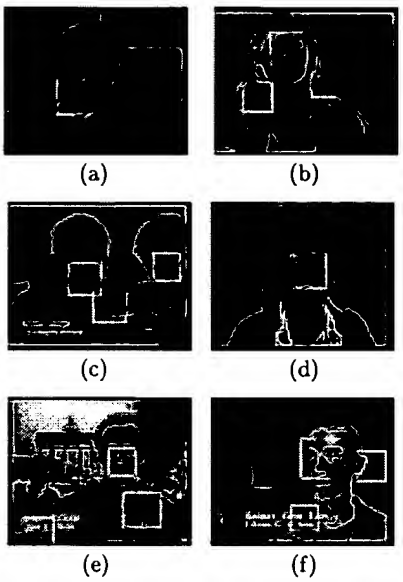


Figure 11: Several detection results of our texture-based face detection algorithm (in the compressed domain)

parameters in the feature vectors experience large variation from sample to sample, which makes them contribute negatively to the separation problem. As indicated by Fig 10, for the specific case of model size 5 block by 5 block, the ideal feature vector length should be 4 to 5 DCT parameters per block, or of about 100 parameters for the entire feature vector.

Based on this observation, we determine the feature vector lengths for all the 6 models in our system, which is listed in Table. 1. Note that the feature vector lengths are all shorter than those used for pixel domain detection. This difference is mainly, as have already been pointed out, due to the block quantization problem in the DCT domain.

### 3.3 Experiments

The texture based face detection algorithm is tested on a variety of pictures, which include: CMU database[6], CMU online face database[7], key frames of news video clips from CNN and NBC as well as pictures downloaded from Internet and scanned from magazines and books. Though coming from different sources and formats, the pictures are all compressed into JPEG format with Adobe Photoshop 5.0 (quality option *medium*, quantizer index 3) before processed. Some detection results are shown in Fig 11.

Compared with the pixel-domain detection algorithms, our algorithm is less accurate because of block quantization, especially the false detection rate is relatively high. This can be seen from Fig 11. To overcome this problem, we seek to combine the texture-based algorithm with a color-based algorithm.

[6]http://www.ius.cs.cmu.edu/IUS/eyes_usr17/har/har1/usr0/har/faces/test/

[7]http://www.ius.cs.cmu.edu/IUS/usrp0/har/FaceDemo/gallery-inline.html

## 4 Combined Color-Based and Texture-Based Face Detection in the DCT Domain

### 4.1 Generating Color Similarity Map

Color-based face detection work in the DCT domain was first discussed by Wang [18]. In this work, our basic design is similar to theirs, except that we do not try to setup a threshold at the color detection stage.

We assume the color pictures in JPEG and MPEG I frames are compressed in $4:2:0$ format. The pictures are partially decoded to restore their DCT parameters in the block structures. The skin color is modeled and detected at the macroblock level. That is, in each macroblock, the DC components of the Cb and Cr blocks are used as the average chromatic feature vector. A Gaussian model $N(\mathbf{v}_s, \mathbf{C}_s)$ for the skin color is created by training over a manually labeled picture database. Based on this skin color model, a color similarity map is generated for each picture at the macroblock level. For macroblock $(i,j)$, if we denote its color feature vector as $\mathbf{v}_{(i,j)}$, then its skin color similarity map entry is

$$\text{color}(i,j) = -\frac{1}{2}\left[\ln 2\pi + \ln|\mathbf{C}_s| + (\mathbf{v}_{(i,j)} - \mathbf{v}_s)^T \mathbf{C}_s^{-1}(\mathbf{v}_{(i,j)} - \mathbf{v}_s)\right] \quad (5)$$

### 4.2 Color Constrained Face Pattern Detection

With the color map available, the face detection problem is further extended from the previous texture domain to the color domain. That is, given an input color picture in its YCbCr format, we can apply the texture-based pattern detection on the Y component and the skin-color map based pattern detection on the CbCr components[8]. Because both detection designs have a statistical expression, it is easy to combine them with a statistical framework, either in a parallel or sequential structure. Though theoretically the parallel structure has the merit of delayed judgment and thus better detection performance, sequential structure is simpler and faster. In addition, the color based detection module is not a balanced module as compared with the texture based detection module in the detection accuracy and reliability sense[9]. Therefore we choose the sequential structure in our system design. A skin color map is at first created by the color analysis module to generate a skin color map as follows. Given a windowed image pattern $\mathcal{W}$, its average skin-color similarity

$$\frac{\sum_{(i,j)\in\mathcal{W}} \text{color}(i,j)}{\sum_{(i,j)\in\mathcal{W}} 1},$$

(where $color(i,j)$ is defined in Eq. 5), is compared with a threshold $T$. Only the windows with an average similarity higher than the threshold are further processed by the texture analysis module as discussed in Section 3.

[8]Because the skin-color similarity map is a map of scalar values, it is straight-forward to design a face pattern detection algorithm based on this similarity map, which should be, in principle, the same as the one we have designed on the texture map.

[9]In this work, we do not spend time to create an example based face pattern on top of the skin-color similarity map as we do in Section 3 on top of the texture map (though it is possible).

| | model size (in block) | model size (in pixel) | masked block numbers | parameters per block | feature vector length |
|---|---|---|---|---|---|
| 1 | 5 | 40 | 23 | 4 | 92 |
| 2 | 6 | 48 | 32 | 3 | 102 |
| 3 | 7 | 56 | 45 | 2 | 90 |
| 4 | 8 | 64 | 60 | 2 | 120 |
| 5 | 9 | 72 | 77 | 2 | 154 |
| 6 | 10 | 80 | 92 | 1 | 92 |

Table 1: Feature vector length at different model sizes

This combination, though simple, is better than both texture-only and color-only algorithms. Compared with texture-only algorithms, the color similarity map offers an additional constraint, which eliminates false alarms in the background without a skin-color appearance as well as improves the processing speed. Compared with color-only algorithms, the texture analysis module helps reduce false alarms introduced by skin-like backgrounds. In addition, the texture analysis module is also useful in locating faces when skin-like backgrounds are close to actual faces, in which case the simple shape analysis modules commonly used in the color-based algorithms always fail.

### 4.3 Experiments

We tested our face detection algorithm based on combined texture and color information over many pictures. The testing picture set we used in Section 3.3 is also used here, except the CMU database and CMU online database are not used because they are grayscale pictures. In Fig. 12, we show some detection results on pictures from various sources.

To evaluate the performance of our algorithm quantitatively, we use the key frames of one day's NBC Nightly News (Feb. 18, 1999, totally 586 frames) as our testing data set. Within this data set, there are 42 faces and 36 of them are frontal and upright ones that are within the coverage of our texture-based face model. The detection performance is listed in Table 2. In Table 2, the new combined texture and color based face detection algorithm (column a) is compared with a color-based detection algorithm (column b). Because the color-based algorithm works in the pixel domain, while the combined texture and color based detection algorithm works in the compressed DCT domain, the comparison is focused on the detection performance, but not the spatial location accuracy of the detected faces.

As we can see from Table 2, the combined texture and color based algorithm has less false alarms than the color-based algorithm because the texture processing module has removed some false alarms introduced by the skin-color backgrounds. In addition, fewer faces are missed by the combined color and texture based algorithm than by the color-based algorithm. This difference is mainly due to the different ability of the two algorithms to detect faces surrounded by skin-like backgrounds. That is, in color-based algorithms, skin-color regions are first detected with a skin-color model based detector and then processed with a shape analysis module. Only the regions have specific aspect ratios are accepted as face regions. When some skin-like backgrounds are close to the actual faces, the detected skin-color regions always get connected, which makes the shape of the detected regions no longer have a face-like shape. In contrast, the texture based approach is not influenced by the backgrounds, as long as they do not *look like* face patterns in the grayscale sense.

| | (a) | (b) |
|---|---|---|
| total faces | 42 | 36 |
| face detected | 34 | 30 |
| face missed | 7 | 5 |
| false alarm | 9 | 4 |
| detection rate | 81% | 83% |
| detection accuracy | 79% | 88% |

Table 2: Performance of the combined color and texture based face detection algorithm

As a result, the combined texture and color based algorithm exhibits better *detection rate* (*i.e.*, the ratio of correctly detected faces v.s. total faces that should be detected) as well as better *detection accuracy* (*i.e.*, the ratio of correctly detected faces v.s. the total detected faces) in this experiment.

In speed performance, the texture analysis module is the most time consuming part in our algorithm. However, its complexity depends on the actually size of the detected skin-color map. On the mentioned CIF size 586 key frames, the average speed of our algorithm is about 1 second per frame on a 366 Celeron PC. We expect to achieve better performance on a better PC with some software optimizations.

## 5 Concluding Remarks

In this paper we mainly developed a texture-based face detection algorithm that works in the compressed DCT domain. This is a new work on compressed domain processing. Our work is based on the previous face detection works designed in the pixel domain, but we discussed major problems we met in the compressed DCT domain, such as block quantization problem, feature vector selection, preprocessing design in the DCT domain, and multi-model based system structure. Due to the block quantization problem, we have to use shorter feature vector than the face detection designs in the pixel domain. Therefore, the proposed texture-based detection algorithm is not as good as its counterparts in the pixel domain. To solve this problem, we proposed to combine the texture-based algorithm with a face color detection algorithm. Experiments indicate

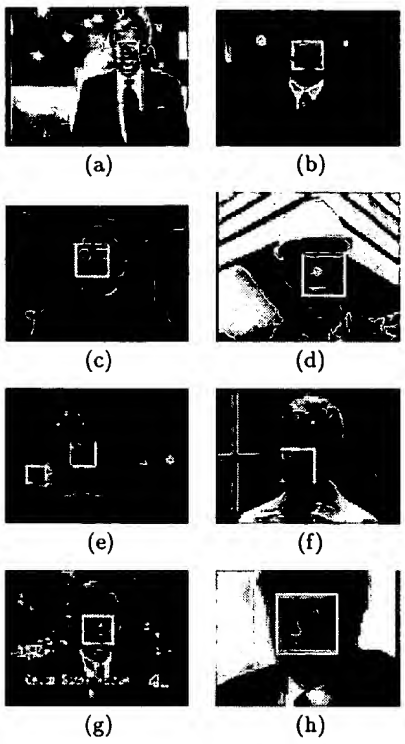


Figure 12: Face detection examples

that the combined texture and color based detection algorithm works better than both texture-only and color-only algorithms.

To sum up, this work is interesting because it first proposed to do traditional pattern detection work on the compressed DCT domain, which is a promising research direction for multimedia content analysis. In addition, in this work we also proposed, for the first time, to combine texture and color information for face detection. This is especially useful for multimedia processing, in which most visual data are in color formats.

## References


[1] M. Abdel-Mottaleb and A. Elgammal. Face detection in complex environments from color images. In *IEEE ICIP*, Kobe, Japan, 1999.

[2] A. Albiol, C.A. bouman, and E.J. Delp. Face detection for pseudo-semantic labeling in video databases. In *IEEE ICIP*, Kobe, Japan, 1999.

[3] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2), 1998.

[4] R. Dugad and N. Ahuja. *A Fast scheme for image size change in the compressed domain*, April 1999. Manual under review.

[5] M. Colobet *et al.* LISTEN: a system for locating and tracking individual speakers. In *Proc. of the 2nd Intl. Conf. on automatic face and gesture recog.*, pages 283–288, Killington, VT, 1996.

[6] K. Fukunaga and W.L.G. Koontz. Application of the karhunen-loeve expansion to feature selection and ordering. *IEEE Trans. on Computers*, 19(4), 1970.

[7] C. Garcia and G. Tziritas. Face detection using quantized skin color regions and wavelet packet analysis. *IEEE Trans. on Multimedia*, 1(3), 1999.

[8] J.B. Lee and B.G. Lee. Transform domain filtering based on piplining structure. *IEEE Trans. on Signal Processing*, 40:2061–2064, 1992.

[9] M. Lew. Information theoretic view-based and modular face detection. In *Proc. of the 2nd Intl. Conf. on automatic face and gesture recog.*, pages 198–203, Killington, VT, 1996.

[10] S. McKenna and S. Gong. Tracking faces. In *Proc. of the 2nd Intl. Conf. on automatic face and gesture recog.*, pages 271–275, Killington, VT, 1996.

[11] N. Merhav and V. Bhaskaran. Fast algorithms for DCT-domain image downsampling and for inverse motion compensation. *IEEE Trans. on CSVT*, 7(3), 1997.

[12] B. Moghadda and A. Pentland. Probalilistic visual learning for object detection. In *Proc. Intl. CVPR*, pages 786–793, Cambridge, MA, 1995.

[13] N. Oliver, A. Pentland, and F. Berard. LAFTER: A real-time lips and face tracker with facial expression recognition. In *Proc. Intl. CVPR*, S.Juan, Puerto Rico, June 1997.

[14] E.E. Osuna, R. Freund, and F. Girosi. Support vector machines: training and applications. Technical report, MIT AI Lab, March 1997. A.I.Memo 1602.

[15] H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. PAMI*, 20(1), 1998.

[16] K.K. Sung. *Learning and Example Selection for Object and Pattern Detection.* PhD thesis, MIT AI lab, 1996. also available as MIT tech. Report AITR 1572.

[17] A.M. Tekalp. *Digital Video Processing.* Prentice Hall, 1996.

[18] H. Wang and S.-F. Chang. A highly efficient system for automatic face region detection in mpeg video. *IEEE Trans. CSVT*, 7(4), 1997.

[19] M.-H. Yang and N. Ahuja. Detecting human faces in color images. In *IEEE ICIP*, Chicago, IL, 1998.

[20] M.-H. Yang and N. Ahuja. Face detection using a mixture of factor analyzers. In *IEEE ICIP*, Kobe, Japan, 1999.